METHOD AND APPARATUS FOR CALCULATING THE REMAINDER OF A MODULO DIVISION

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a method and apparatus for calculating the remainder of a modulo division, and more particularly, to a non-iterative technique for calculating the remainder of a modulo division.

DESCRIPTION OF THE RELATED ART

Most computer systems require the value modulo n of an integer m, generally written m mod n. M mod n operations are utilized in a number of computer processes, for example, address generating, communication channel load balancing, computer graphics, telephone switching and telephone transmission, packet switching and transmission, and digital message encoding/decoding. Therefore, a routine that accepts integers m and n and produces m mod n is a useful computer tool.

The traditional method of calculating the remainder of modulo division is an iterative algorithm:

T,0025

$$\frac{N}{D} = Q + \frac{R}{D} \tag{1}$$

where

N is the dividend,

D is the divisor,

Q is the quotient,

R is the remainder, and

$0 \leq R < D$.

Assuming Q is an n-bit positive integer, the value of Q can be denoted as a sum of binary positions:

T,0024

$$Q = \sum_{j=0}^{n-1} Q_j \cdot 2^j \tag{2}$$

where Q_j is the jth bit of $Q, Q_j = \{0,1\}$ for all j

2

The traditional iterative algorithm begins by initializing R to N, i.e., R(n)=N and initializing Q_n to 1 since N is a positive integer. Then the traditional algorithm calculates R(j) and Q_j for j=(n-1) to j=0 iteratively. The complete algorithm, for dividing an n-bit positive integer N by an n-bit positive integer D resulting in an n-bit positive quotient Q and n-bit positive remainder R, is shown in Figure 1.

An example of the traditional iterative algorithm is set forth below:

Example 1:

Assuming N=25, D=7, and n=3 $\Rightarrow$ 25 mod 7 = 4

step A1:	R(3)=25, Q_3=1, j=2
step B1:	j<0 ? No
step C1:	Q_3=1 ? Yes
step D1:	$R(2)=R(3)-D2^j=25-(7)(4)=-3$
step E1:	R(2)<0 ? Yes
step F1:	$Q_j=Q_2=0$
step G1:	j=j-1=2-1=1
step B2:	j<0 ? No
step C2:	Q_2=1 ? No
step H1:	$R(1)=R(2)+D2^j=-3+(7)(2)=11$
step E2:	R(1)<0 ? No
step I1:	Q_1=1
step G2:	j=j-1=1-1=0
step B3:	j<0 ? No
step C3:	Q_1=1 ? Yes
step D2:	$R(0)=R(1)-D2^j=11-(7)(1)=4$
step E3:	R(0)<0 ? No
step I2:	Q_0=1
step G3:	j=j-1=0-1=-1
step B4:	j<0 ? Yes
step J1:	$R = R(-1)=R(0)+(1-Q_0)\cdot D=4+(1-1)7=4$

According to the traditional iterative algorithm, as illustrated in Figure 1 and Example 1, the remainder of 25 mod 7 is correctly computed to be 4.

The traditional iterative algorithm for determining the remainder of modulo division is a general purpose method, in that any two positive integers may be entered for N and D. However, as illustrated in Example 1, the traditional iterative algorithm requires numerous computations due to its iterative nature. For example, if D is an n-bit integer with a value $D=2^{n-1}$ and n is any positive integer less than or equal to $(D-1)^2$, i.e., n is a $2 \cdot n$-bit integer at maximum, the traditional iterative method requires $6 \cdot n$ numbers of condition testing, $2 \cdot n$ numbers of multiplications (or shifts) and $2 \cdot n$ additions. Therefore, the total number of operations required is $10 \cdot n$, excluding any value-assigning operations.

SUMMARY OF THE INVENTION

The present invention is directed to a method and apparatus for calculating the remainder of a modulo division. The present invention is directed to a non-iterative technique for calculating the remainder of modulo division. The present invention requires significantly fewer operations than the traditional iterative technique for the same calculation. Furthermore, the number of calculations requires in the present invention is independent of the number of bits of the divisor in the modulo operation.

Two requirements of the non-iterative technique of the present invention are that the value of the divisor D should be equal to 2^n-1 (where n is the number of bits of the divisor D) and the value of the dividend N should be less than or equal to $(D-1)^2$, but great than or equal to zero.

Because the two constraints set forth above are basic constraints of Reed-Solomon coding, the present invention is extremely useful in applications that implement Reed-Solomon coding. Reed-Solomon coding involves algebraic operations in a Galois field. Reed-Solomon coding is a type of forward-error correcting coding that is used extensively in data communications. In Reed-Solomon coding, both conditions of the technique of the present invention are met and the algorithm of the present invention greatly improves the Reed-Solomon coding speed. In particular, the present invention is at least four times faster than the traditional iterative algorithm for a 16-bit fixed-point digital signal processor with special instructions supporting iterative division. Furthermore, if the 16-bit

fixed-point DSP has no special division instruction, the algorithm of the present invention is at least twelve times faster than the traditional iterative algorithm.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a flowchart of the traditional iterative method for determining the remainder of a modulo division.

Figure 2 illustrates a hardware arrangement of the present invention of one embodiment.

Figure 3 illustrates a flowchart of the technique of the present invention for determining the remainder of a modulo division.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 2 illustrates an apparatus of the present invention in one embodiment. In particular, Figure 2 illustrates a coder 10, which includes a processing unit 12, which implements the algorithm of the present invention. The processing unit 12, which performs the algorithm of the present invention, may receive a computer program to implement the algorithm of the present invention via an article of manufacture 14 or propagated signal 20. The article of manufacture 14 further includes a medium 16, in addition to the computer program 18.

The processing unit 12 may be any analog or digital processor, and either hardwired or software programmable to carry out the algorithm of the present invention. Processing unit 12 could be a general purpose processor, a digital processor (DSP), an algorithmic logic unit (ALU), or any other processing element, either discrete or integrated, which performs the algorithm to be described below. The coder 10 could be any type of coder which includes the processing unit 12, such as, for example, a Reed-Solomon coder.

The processing unit 12 implements a non-iterative technique for calculating the remainder of modulo division. The number of calculations performed by the processing unit 12 is independent of the number of bits of the divisor in the modulo operation. Two requirements of the non-iterative algorithm of the present invention are that the value of the divisor D should be equal to 2^n-1 (where n is the number of bits of the divisor D) and

the value of the dividend N should be less than or equal to $(D-1)^2$, but greater than or equal to 0. If the two conditions set forth are met, the remainder R of M mod D is determined by

T,0060
T,0061

summing the upper $\lceil \frac{n}{2} \rfloor$ and lower $\lceil \frac{n}{2} \rfloor$ bits of the dividend N to produce the remainder R. The two conditions described above enable the processing unit 12 to determine the remainder of a modulo division operation with significantly fewer operations than the traditional iterative technique for the same calculation.

The algorithm of the present invention will now be described in more detail.

For a given dividend N and divisor D, computing N/D gives a quotient Q and remainder R. The relation between these quantities is:

T,0062

$$\frac{N}{D} = Q + \frac{R}{D} \tag{3}$$

where $0 \leq R < D$

Rearrange the equation above gives:

$$N = Q \cdot D + R \tag{4}$$

Equation 4 can be written as

T,0063

$$N = \begin{cases} Q \cdot (D+1) + (R-Q) & \text{if } R \geq Q \\ (Q-1) \cdot (D+1) + [(R-Q)+(D+1)] & \text{else} \end{cases} \tag{5}$$

Equation 5 can be reduced to

$$N = Q' \cdot (D+1) + R' \tag{6}$$

where,

T,0064

$$Q' = \begin{cases} Q & \text{if } R \geq Q \\ (Q-1) & \text{else} \end{cases} \tag{7}$$

$$R' = \begin{cases} (R-Q) & \text{if } R \geq Q \\ [(R-Q)+(D+1)] & \text{else} \end{cases}$$

Comparing Equation 4 and Equation 6 it can be seen that Q' and R' are the quotient and remainder of N(mod (D+1)) and Equation 7 gives the relationship between Q, R and Q', R'. Adding Q' and R' gives:

T,0065

$$Q' + R' = \begin{cases} R & \text{if } R \geq Q \\ R + D & \text{else} \end{cases} \tag{8}$$

6

Equation 8 shows the sum of Q' and R' can be equal to R or R + D. Given R is a nonnegative integer less than D as shown in Equation 3, R can be obtained by comparing the sum of R' and Q' with D:

T,0070

$$R = \begin{cases} Q'+R' & \text{if } Q'+R' < D \\ Q'+R'-D & \text{else} \end{cases} \tag{9}$$

If the quotient Q' and remainder R' of N(mod(D+1)) are available, the remainder of N(mod D) can be obtained using Equation (9).

Given the above conclusion, now the question is whether there is an efficient way to calculate the quotient and remainder of N(mod (D+1)). Assuming D=2^n-1,

T,0071

$$N = \sum_{j=0}^{2\cdot n-1} a_j \cdot 2^j \quad \text{where } a^j \text{ is the } j\text{th bit of } N, a^j = \{0,1\} \text{ for all } j$$

then (D+1) = 2^n. This fact results in a quick method of obtaining Q' and R'. Assuming N≤$(D-1)^2$ means N is less than or equal to 2·n bits. The binary representation of N is:

T,0072

$$N = \left(\sum_{k=0}^{n-1} a_{k+n} \cdot 2^k\right) \cdot 2^n + \sum_{j=0}^{n-1} a_j \cdot 2^j \tag{10}$$

where

T,0073

$$\sum_{k=0}^{n-1} a_{k+n} \cdot 2^k < 2^n \text{ and } \sum_{j=0}^{n-1} a_j \cdot 2^j < 2^n \tag{11}$$

Comparing Equation 6 and Equation 10 and assuming 2^n=(D+1), the quotient Q' and remainder R' in Equation 6 can be obtained as:

T,0074

$$Q' = \sum_{k=0}^{n-1} a_{k+n} \cdot 2^k \quad \text{where } a_{k+n} \text{ is the } (n+k)\text{th bit of } N, a_{k+n} = \{0,1\} \text{ for all } k$$

and

$$R' = \sum_{j=0}^{n-1} a_j \cdot 2^j \quad \text{where } a_j \text{ is the } j\text{th bit of } N, a_j = \{0,1\} \text{ for all } j \tag{12}$$

Equation (12) means that the quotient Q' and remainder R' of N(mod (D+1)) are the numbers made up by the high and low n bits of N, respectively.

A flow chart of the present invention is illustrated in Figure 3. Example 2, set forth below, is explained in conjunction with the flowchart illustrated in Figure 3.

Example 2:

If N=25 and n=3,

T, 0080

Condition 1: $D=2^n-1=7 \Rightarrow$ True

Condition 2: $0 \leq N \leq (D-1)^2$

$0 \leq 25 \leq 36 \Rightarrow$ True

Assuming N=25, D=7, and n=3 $\Rightarrow$ 25 mod 7=4.

N=25 is represented as a 2n bit number (five bits with a leading zero added)

011⁝001

T, 0081

step A': a = high bits of N $\rightarrow$ a = 011

step B': b = low n bits of N $\rightarrow$ b = 001

step C': c = a + b = 100

step D': c < D ? Yes

step E': N(mod D) = c = 100 = 4

According to the technique of the present invention, the remainder of 25 mod 7 is correctly computed to be 4. Comparing the present invention to the traditional iterative technique, the present invention only requires 5 operations for determining the remainder R, as illustrated in Example 2, whereas the traditional iterative technique, as illustrated in Example 1, requires 21 operations.

Another example illustrates the additional processing performed in step F' of the present invention.

Example 3:

If N=15 and n=3,

T, 0082

Condition 1: $D = 2^3-1=7 \Rightarrow$ True

Condition 2: $0 \leq N \leq (D-1)^2$

$0 \leq 15 \leq 36 \Rightarrow$ True

Assuming N=15, D=7, and n=3 $\Rightarrow$ 15 mod 7 = 1

N=15 is represented as a 2n bit number (four bits with two leading zeroes added)

001⁝111

T,0090

step A': a = high n bits of N → a = 001

step B': b = low n bits of N → b = 111

step C': c = a + b = 1000

step D': 1000 < 0111 ? No

step F': c = c-D = 1000-0111 = 0001

step E': N (mod D) = c = 0001 = 1

Again, according to the technique of the present invention, the remainder of 15 mod 7 is correctly computed to be 1.

The invention being thus described, it will be obvious that the same may be varied in many ways. Such variations are not to be regarded as a departure from the spirit and scope of the invention, and all such modifications as would be obvious to one skilled in the art are intended to be included within the scope of the following claims.